1.11 — Developing your first program

ALEX[1] MARCH 12, 2024

The preceding lessons have introduced a lot of terminology and concepts that we'll use in just about every program we create. In this lesson, we'll walk through the process of integrating this knowledge into our first simple program.

Multiply by 2

First, let's create a program that asks the user to enter an integer, waits for them to input an integer, then tells them what 2 times that number is. The program should produce the following output (assume I entered 4 as input):

```
Enter an integer: 4
Double that number is: 8
```

How do we tackle this? In steps.

Best practice

New programmers often try to write an entire program all at once, and then get overwhelmed when it produces a lot of errors. A better strategy is to add one piece at a time, make sure it compiles, and test it. Then when you're sure it's working, move on to the next piece.

We'll leverage that strategy here. As we go through each step, type (don't copy/paste) each program into your code editor, compile, and run it.

First, create a new console project.

Now let's start with some basic scaffolding. We know we're going to need a main() function (since all C++ programs must have one), so if your IDE didn't create a blank one when you created a new project, let's create one:

```
1 int main()
2 {
3     return 0;
4 }
```

We know we're going to need to output text to the console, and get text from the user's keyboard, so we need to include iostream for access to std::cout and std::cin.

```
#include <iostream>

int main()
{
    return 0;
}
```

Now let's tell the user that we need them to enter an integer:

```
#include <iostream>

int main()
{
    std::cout << "Enter an integer: ";

    return 0;
}
```

At this point, your program should produce this result:

```
Enter an integer:
```

and then terminate.

Next, we're going to get the user's input. We'll use std::cin and `operator>>` to get the user's input. But we also need to define a variable to store that input for use later.

```
#include <iostream>

int main() // note: this program has an error somewhere
{
    std::cout << "Enter an integer: ";

    int num{ }; // define variable num as an integer variable
    std::cin << num; // get integer value from user's keyboard

    return 0;
}
```

Time to compile our changes... and...

Uh oh! Here's what the author got on Visual Studio 2017:

```
1>------ Build started: Project: Double, Configuration: Release Win32 ---
1>Double.cpp
1>c:\vcprojects\double\double.cpp(8): error C2678: binary '<<': no operat
1>c:\vcprojects\double\double.cpp: note: could be 'built-in C++ operator<
1>c:\vcprojects\double\double.cpp: note: while trying to match the argume
1>Done building project "Double.vcxproj" -- FAILED.
========== Build: 0 succeeded, 1 failed, 0 up-to-date, 0 skipped ========
```

We ran into a compile error!

First, since the program compiled before we made this latest update, and doesn't compile now, the error *must* be in the code we just added (lines 7 and 8). That significantly reduces the amount of code we have to scan to find the error. Line 7 is pretty straightforward (just a variable definition), so the error probably isn't there. That leaves line 8 as the likely culprit.

Second, this error message isn't very easy to read. But let's pick apart some key elements: The compiler is telling us it ran into the error on line 8. That means the actual error is probably on line 8, or possibly the preceding line, which reinforces our previous assessment. Next, the compiler is telling you that it couldn't find a '<<' operator that has a left-hand operand of type std::istream (which is the type of std::cin). Put another way, operator<< doesn't know what to do with std::cin, so the error must be either with our use of std::cin or our use of operator<<.

See the error now? If you don't, take a moment and see if you can find it.

Here's the program that contains the corrected code:

```
1  #include <iostream>
2
3  int main()
4  {
5      std::cout << "Enter an integer: ";
6
7      int num{ };
8      std::cin >> num; // std::cin uses operator >>, not operator <<!
9
10     return 0;
11 }
```

Now the program will compile, and we can test it. The program will wait for you to enter a number, so let's enter 4. The output should look like this:

```
Enter an integer: 4
```

Almost there! Last step is to double the number.

Once we finish this last step, our program will compile and run successfully, producing the desired output.

There are (at least) 3 ways we can go about this. Let's go from worst to best.

The not-good solution

```
#include <iostream>

// worst version
int main()
{
    std::cout << "Enter an integer: ";

    int num{ };
    std::cin >> num;

    num = num * 2; // double num's value, then assign that value back to num

    std::cout << "Double that number is: " << num << '\n';

    return 0;
}
```

In this solution, we use an expression to multiply *num* by 2, and then assign that value back to *num*. From that point forward, *num* will contain our doubled number.

Why this is a bad solution:

- Before the assignment statement, num contains the user's input. After the assignment, it contains a different value. That's confusing.
- We overwrote the user's input by assigning a new value to the input variable, so if we wanted to extend our program to do something else with that input value later (e.g. triple the user's input), it's already been lost.

The mostly-good solution

```
#include <iostream>

// less-bad version
int main()
{
    std::cout << "Enter an integer: ";

    int num{ };
    std::cin >> num;

    int doublenum{ num * 2 }; // define a new variable and initialize it
with num * 2
    std::cout << "Double that number is: " << doublenum << '\n'; // then
print the value of that variable here

    return 0;
}
```

This solution is pretty straightforward to read and understand, and resolves both of the problems encountered in the worst solution.

The primary downside here is that we're defining a new variable (which adds complexity) to store a value we only use once. We can do better.

The preferred solution

```
#include <iostream>

// preferred version
int main()
{
    std::cout << "Enter an integer: ";

    int num{ };
    std::cin >> num;

    std::cout << "Double that number is: " <<  num * 2 << '\n'; // use an expression to multiply num * 2 at the point where we are going to print it

    return 0;
}
```

This is the preferred solution of the bunch. When std::cout executes, the expression *num * 2* will get evaluated, and the result will be double *num*'s value. That value will get printed. The value in *num* itself will not be altered, so we can use it again later if we wish.

This version is our reference solution.

Author's note

The first and primary goal of programming is to make your program work. A program that doesn't work isn't useful regardless of how well it's written.

However, there's a saying I'm fond of: "You have to write a program once to know how you should have written it the first time." This speaks to the fact that the best solution often isn't obvious, and that our first solutions to problems are usually not as good as they could be.

When we're focused on figuring out how to make our programs work, it doesn't make a lot of sense to invest a lot of time into code we don't even know if we'll keep. So we take shortcuts. We skip things like error handling and comments. We sprinkle debugging code throughout our solution to help us diagnose issues and find errors. We learn as we go -- things we thought might work don't work after all, and we have to backtrack and try another approach.

The end result is that our initial solutions often aren't well structured, robust (error-proof), readable, or concise. So once your program is working, your job really isn't done (unless the program is a one-off/throwaway). The next step is to cleanup your code. This involves things like: removing (or commenting out) temporary/debugging code, adding comments, handling error cases, formatting your code, and ensuring best practices are followed. And even then, your program may not be as simple as it could be -- perhaps there is redundant logic that can be consolidated, or multiple statements that can be combined, or variables that aren't needed, or a thousand other little things that could be simplified. Too often new programmers focus on optimizing for performance when they should be optimizing for maintainability.

Very few of the solutions presented in these tutorials came out great the first time. Rather, they're the result of continual refinement until nothing else could be found to improve. And in many cases, readers still find plenty of other things to suggest as improvements!

All of this is really to say: don't be frustrated if/when your solutions don't come out wonderfully optimized right out of your brain. That's normal. Perfection in programming is an iterative process (one requiring repeated passes).

Author's note

One more thing: You may be thinking, "C++ has so many rules and concepts. How do I remember all of this stuff?".

Short answer: You don't. C++ is one part using what you know, and two parts looking up how to do the rest.

As you read through this site for the first time, focus less on memorizing specifics, and more on understanding what's possible. Then, when you have a need to implement something in a program you're writing, you can come back here (or to a reference site) and refresh yourself on how to do so.

Quiz time

Question #1

Modify the solution to the "best solution" program above so that it outputs like this (assuming user input 4):

```
Enter an integer: 4
Double 4 is: 8
Triple 4 is: 12
```

Hide Solution (javascript:void(0))[2]

```
#include <iostream>

int main()
{
    std::cout << "Enter an integer: ";

    int num{ };
    std::cin >> num;

    std::cout << "Double " << num << " is: " << num * 2 << '\n';
    std::cout << "Triple " << num << " is: " << num * 3 << '\n';

    return 0;
}
```

Next lesson

1.x Chapter 1 summary and quiz

3

Back to table of contents

4

Previous lesson

1.10 Introduction to expressions

5

6

B U URL INLINE CODE C++ CODE BLOCK HELP!

Leave a comment...

Name*

Email*

Notify me about replies:

POST COMMENT

Find a mistake? Leave a comment above!

Avatars from https://gravatar.com/[9] are connected to your provided email address.

280 COMMENTS Newest

sajLMAO
April 14, 2024 3:41 pm

Finally first exercise so far in this course :)

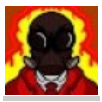
Last edited 16 hours ago by sajLMAO

0 Reply

Kenneth.O
April 13, 2024 12:23 pm

```cpp
#include <iostream>

using namespace std;

int main()
{
    cout << "Enter an integer: " ;

   int num{};
   cin >> num;
   int doublenum{num * 2};

   cout << "Double " << num << " is " << doublenum <<'\n'; // calling the
variable 'doublenum';

   cout << "Triple " << num << " is " << num * 3 << '\n'; // directly doing
the expression on the same line;

    // both work as intended;


    return 0;
}
```

0 Reply



Kojo Bailey
April 3, 2024 6:41 am

It's so cool seeing people post their code as they go along the lessons lol

A good reminder that there are always many of us out there learning C++

0 Reply

yoyoyo47
March 28, 2024 7:28 am

I did go a little bit further

```cpp
#include <iostream>

int main()
{
    while (true) {
        char yo;
        float num1, num2;

        std::cout << "Wtf you wanna to do? (*, /, +, -): ";
        std::cin >> yo;

        std::cout << "Enter TwO bIg NumBaaa: ";
        std::cin >> num1 >> num2;


        switch (yo)
        {
        case '*':
            std::cout << num1 * num2 << '\n' << '\n';
            break;

        case '/':
            if (num2 != 0)
                std::cout << num1 / num2 << '\n' << '\n';
            else
                std::cout << "You're bOzO or whAt? You cannot divide by
ZERO" << '\n' << '\n';
            break;

        case '+':
            std::cout << num1 + num2 << '\n' << '\n';
            break;

        case '-':
            std::cout << num1 - num2 << '\n' << '\n';
            break;

        default:
            std::cout << "WTF is that?" << '\n' << '\n';
        }
    }

        return 0;
    }
```

Last edited 18 days ago by yoyoyo47

2 Reply

Kojo Bailey

Reply to yoyoyo47 [10] April 3, 2024 6:38 am

Note: You can just `"\n\n"` instead of `'\n' << '\n'`.

You can also combine strings together to have `"WTF is that?\n\n"`, although that's more up to you. I'd even consider having `"\n\n"` in a `newline` variable or something.

```
const std::string newline = "\n\n"; // Can change later to adjust spacing if wanted.
std::cout << "WTF is that?" << newline;
```

Also, nitpicking that you should have a newline before the `while` opening brace for formatting consistency:

```
while (true)
{
```

0 Reply

faraz
March 25, 2024 8:02 pm

#include <iostream>

int main()
{
int num{};

std::cout << "Enter a number you would like to have doubled \n";
std::cin >> num;

std::cout << "Double your number " << num << "\nIs " << num * 2 << std::endl;

std::cout << "Triple your number " << num << "\nIs " << num * 3;

return 0;
}

1 Reply

Hanhho
March 18, 2024 6:57 am

#include <iostream>
using namespace std;

int main{
int a, b, c;
cout << "Nhập một số nguyên : ";
cin >> a;
cout << "Đôi " << a << " là : " << a*2;
cout << "Bộ ba " << a << "là : " << a*3;

return 0;
}

1 Reply

Sir_Jason
March 18, 2024 3:47 am

```
#include "stdafx.h" // includes "iostream" and "using namespace std"

int main() {
    int num{ }, mnum{ };
    cout << "Enter an integer: \n";
    cin >> num;
    cout << "By how much should we multiple it?\n";
    cin >> mnum;
    cout << "Your number is: " << num*mnum;
}
```

Last edited 28 days ago by Sir_Jason

0 Reply

Kamil G.
March 17, 2024 9:43 am

My code:

```cpp
#include <iostream>

int main(){

    std::cout << "Enter the intager: ";

    int num {   };

    std::cin >> num;

    // The not-good solution:
    /*
    num = num * 2;
    std::cout << "Double that number is: " << num;
    */

    // The mostly-good solution:
    /*
    int doubleNum { num * 2 };

    std::cout << "Double that number is: " << doubleNum;
    */

    // The prefered solution:
    /*
    std::cout << "Double that number is: " << (num * 2) << '\n';

    // + triple it:
    std::cout << "Triple that number is: " << (num * 3);
    */

    // Or like that:
    std::cout << "Double " << num << " is: " << (num * 2) << '\n'
                    << "Triple " << num << " is: " << (num * 3);

    return 0;
}
```

0 Reply

Newchallenger147
March 13, 2024 4:15 am

std::cout << "Enter an integer: ";

int x{};

std::cin >> x;

int y{ x * 2 };
std::cout << "Double 4 is: " << y << '\n';

int z{ y * 3 };
std::cout << "Triple 4 is: " << z;

return 0;

was this fine?? or should improve more..

0 Reply

Kojo Bailey

Reply to Newchallenger147 [11] April 3, 2024 6:53 am

Here are some simple changes I'd suggest, although you might want to ignore some of these things depending on what you're trying to do.

```
#include <iostream>

int main() {
    int x{}; // Initialised `x` before asking for input.
    std::cout << "Enter an integer: ";
    std::cin >> x;
    std::cout << "\n\n"; // Added some spacing.

    /*
        - Made text more descriptive (not sure why it was "4"
before).
        - Removed variable defintions so the ouputs are just
calculated directly.
        - Added final newline after the final output (good
practice).
    */
    std::cout << "Double your integer is: " << x * 2 << '\n';
    std::cout << "Triple your integer is: " << x * 3 << '\n';
}
```

0 Reply

Caleb

March 11, 2024 8:04 am

this is mine, anyway i could make it better or is it fine

#include <iostream>

int main()

{
std::cout << "Enter the integer , I will double it then triple it. \n ";

int anumber{};
std::cin >> anumber;

std::cout << "Double " << anumber << " is " << anumber * 2 << " and triple " << anumber << " is " << anumber * 3;
return 0;

}

1 Reply

Eric
March 5, 2024 3:51 pm

Added in a while loop to ask if the user would like to continue inputting new values

1 Reply

IceFloe
February 18, 2024 6:53 pm

and finally I compiled a full-fledged program, where in fact we worked out the theory, of course this is just the beginning of the path, it will be much more interesting further, because you can come up with conditions yourself to practice

4 Reply

GigaDawn
February 15, 2024 12:03 am

why does in terminal the result show

enter an integer: 5

double of that number is:102592

0 Reply

> **GigaDawn**
> Reply to GigaDawn [12] February 15, 2024 12:15 am
>
> nevermind i found the issue, i type '\n ' instead of '\n' but i do wonder why does that happened?
>
> *Last edited 2 months ago by GigaDawn*
>
> 0 Reply
>
> > **Alex** Author
> > Reply to GigaDawn [13] February 16, 2024 10:54 am
> >
> > Because `/n` is a multicharacter literal, and multicharacter literals unexpectedly output as numeric values.
> >
> > 1 Reply

Eric

Reply to Alex [14] March 8, 2024 8:10 am

This might be good to add to the actual lesson as a warning. I think its a common mistake a lot of people might make on their first go around that you can link back to a previous lesson!

0 Reply

Alex Author

Reply to Eric [15] March 11, 2024 12:04 pm

It's already mentioned in previous lesson https://www.learncpp.com/cpp-tutorial/introduction-to-iostream-cout-cin-and-endl/

0 Reply

Cameron

February 5, 2024 12:12 pm

```
#include <iostream>

using namespace std;

int main() {
int userNum;

cout << "Please enter a number:" << endl;

cin >> userNum;

cout << "Double " << userNum << " is: " << userNum * 2 << endl;

cout << "Triple " << userNum << " is: " << userNum * 3 << endl;

return 0;
}
```

0 Reply

Jonathan

February 5, 2024 5:45 am

Directly copying code from lesson my compiler (c++ compiler app for Android) states :

Expected unqualified input by num

Is this a problem with my compiler.

As stated I have directly copypasta the code from best solution

Thanks

J

0 Reply

GigaMau
February 3, 2024 2:24 am

Code using functions (with extra bonus function to multiply by user input)

```
#include <iostream>

int doubled(int number) {
int doubledNum = number * 2;
return doubledNum;
}
int tripled(int number) {
int tripledNum = number * 3;
return tripledNum;
}
int multiply(int number, int multiplyer) {
int multNum = number * multiplyer;
return multNum;
}

int main() {
int number{};
std::cout << "please enter number: ";
std::cin >> number;
std::cout << "Double " << number << " is " << doubled(number) <<"\n";
std::cout << "Trible " << number << " is " << tripled(number)<<"\n";
// use multiply function
std::cout << "Do you want to multiply by a specific number? (y/n): ";
char answer{};
std::cin >> answer;
if (answer == 'y') {
int multiplyer{};
std::cout << "please enter integer: ";
std::cin >> multiplyer;
std::cout << "the number is " << multiply(number, multiplyer);
}
//I don't know how to use if-else statements yet just a test so there's no else
```

return 0;
}

1 Reply

Peter
January 25, 2024 12:39 am

```
#include <iostream>


int main()
{

    std::cout << "*DOUBLE-TRIPLE CALCULATOR*" << '\n';

    float NumFromUser;

    std::cout << "Please type your number and press Enter: ";

    std::cin >> NumFromUser;

    std::cout << "Double " << NumFromUser << " is: " << NumFromUser * 2 <<
'\n';
    std::cout << "Triple " << NumFromUser << " is: " << NumFromUser * 3 <<
'\n';

    return 0;
}
```

I Just wondering is it recommended to use the `{}` at the end of `float NumFromUser`?

Last edited 2 months ago by Peter

0 Reply

Ajit Mali
Reply to Peter [16] January 25, 2024 3:09 am

yes as it is best practice

0 Reply

Peter Iliev
Reply to Ajit Mali [17] January 25, 2024 8:30 am

Yes, I guess so, but is it there any technical reason like better reading from compiler or other computation?
When we put empty braces there is a sign that this is a value initialization. Something that says it's with initial empty (or 0) value and will be filled later.

I understand that it is a best practice to use, just wandering is it for human side as a better reading the code, or could case any warnings in some cases?

0 Reply

Ajit Mali

Reply to Peter Iliev [18] January 25, 2024 8:43 am

when I just use
`float x`
the compiler gives a warning, variable 'x' is uninitialized and
as for the integer
int x = 3.5; // x will be 3 as 0.5 will be dropped
and int x{ 3.5 }; will raise an error

that is it's better to use list initializer

and to be consistent you should always use this.

Last edited 2 months ago by Ajit Mali

0 Reply

DivideandConquer

January 24, 2024 3:23 pm

I love how you covered iterative development with such elegance! Not many tutorials cover how to start a code project. I love how you cover software engineering and computer science in a way that is easy to understand. Thank you!! You are a blessing!

0 Reply

Vladyslav

December 24, 2023 6:18 am

```cpp
#include <format>
#include <iostream>

int main() {
    std::cout << "Enter an integer to be multiplied: ";
    int x{};
    std::cin >> x;

    std::cout << "Enter a multiplier: ";
    int multiplier{};
    std::cin >> multiplier;

    std::cout << std::format("{} * {} = {}", x, multiplier, x * multiplier);

    return 0;
}
```

A little program that multiplies an integer by any other integer, not just 3. Also uses `std::format` from the C++20 standard to get rid of my own ugly formatting :)

5 Reply

AbeerOrTwo
December 14, 2023 6:56 pm

These practices are pretty helpful ngl

0 Reply

Links

1. https://www.learncpp.com/author/Alex/
2. javascript:void(0)
3. https://www.learncpp.com/cpp-tutorial/chapter-1-summary-and-quiz/
4. https://www.learncpp.com/
5. https://www.learncpp.com/cpp-tutorial/introduction-to-expressions/
6. https://www.learncpp.com/developing-your-first-program/
7. https://www.learncpp.com/cpp-tutorial/introduction-to-literals-and-operators/
8. https://www.learncpp.com/cpp-tutorial/function-return-values-value-returning-functions/
9. https://gravatar.com/
10. https://www.learncpp.com/cpp-tutorial/developing-your-first-program/#comment-595188
11. https://www.learncpp.com/cpp-tutorial/developing-your-first-program/#comment-594622
12. https://www.learncpp.com/cpp-tutorial/developing-your-first-program/#comment-593643

13. https://www.learncpp.com/cpp-tutorial/developing-your-first-program/#comment-593644
14. https://www.learncpp.com/cpp-tutorial/developing-your-first-program/#comment-593728
15. https://www.learncpp.com/cpp-tutorial/developing-your-first-program/#comment-594444
16. https://www.learncpp.com/cpp-tutorial/developing-your-first-program/#comment-592837
17. https://www.learncpp.com/cpp-tutorial/developing-your-first-program/#comment-592846
18. https://www.learncpp.com/cpp-tutorial/developing-your-first-program/#comment-592854